A System of Suppositional Logic
as Embodied in the Proof Checker Mizar MSE

PART ONE: ON JASKOWSKI AND EUCLID

1. Historical preliminaries

Jan Lukasiewicz and Stanislaw Jaskowski are credited with being the first who remarked that practising mathematicians do not appeal to logical theories but make use of other methods of reasoning [see Footnote]. They meant those logical theories which were worked out by Russell, Frege, Peano, Hilbert, and Polish logicians, including Lukasiewicz himself. According to Jaskowski’s testimony, in 1926 Lukasiewicz raised the problem of how to build a logical system that would agree with the practice of employing suppositions in proving theorems. In 1934 Jaskowski offered a solution of this problem in his study On the Rules of Supposition in Formal Logic. That study yielded a paradigm of what has been called suppositional logic.

In the same year 1934, two authors, independently of each other, initiated the method of natural deduction: apart from Stanislaw Jaskowski, it was Gerhard Gentzen. The latter expressed his motivation in the identical way, when he wrote: "We wish to set up a formalism which would represent as nearly as possible the actual logical inferences occurring in mathematical proofs." (Gentzen 1934, p.34; the English quotation after Marciszewski 1981). His system NK is elegant for its two symmetric sets of rules, those for the introduction of logical constants, and those for the elimination of constants. Moreover, his sequent calculus offers Hauptsatz - the famous cut-elimination theorem called also normal form theorem which contributed so much to the development of 20th century logic, also in its computerized form. This may account for the renown won by Gentzen systems.

The term "natural deduction" does not appear in Jaskowski’s paper; instead, he uses the phrase "the method of suppositions". It was Gentzen 1934-35 who introduced the term "das natuerliche Schliessen". The first communication on suppositional proofs appeared in the proceedings of the first Polish Mathematical Congress, 1927, entitled Księga pamiątkowa pierwszego polskiego zjazdu matematycznego, Krakow 1929; in a note Jaskowski reported on his results presented at Lukasiewicz’s seminar in 1926.

Jaskowski’s tradition was continued in Poland, especially in the textbook Slupecki and Borkowski 1967, and other works by either of these authors; let both the book in question and the system it contains be referred to as SB. There is a remark in SB that it belongs to natural deduction systems as invented by Jaskowski and by Gentzen, and that it differs from each of them in some details. In fact, though SB resembles Gentzen in the division of inference rules into the introduction set and the elimination set, and it resembles Jaskowski in a method of structuring proofs, it has some original features which are something more than insignificant details.

There is a feature in Jaskowski’s system which did not reappear in SB, and which so far remains fairly unnoticed, in spite of its being closely related to the idea of rendering mathematical practice with a logical theory. Let us discuss it briefly in the next section.

2. Whether natural deduction may become more natural

Though the method of natural deduction as found in Gentzen, in SB, etc, brings us closer to actual methods of reasoning, there is something in it that can hardly be called natural. It is usual in such systems that at the right margin of each proof line one makes reference to both the premisses from which this line derives and the rule which justifies the derivation. This usage is very convenient, indeed, and may be specially recommended for didactic purposes. However, if a formalization is to reconstruct natural inferences as truly as possible, the method of referring to rules should somehow be altered. This requires a new kind of formalization in which the use of inference rules would more resemble those linguistic means which we have in the mathematical vernacular.
The need for such a formalization might not have been felt until computers proved ready for either simulating or checking our intellectual activities, including the proving of theorems. Once such possibilities have opened, one may expect a device for writing mathematical texts in the familiar traditional form, as best suiting our habits, the same form being logically verifiable with a computerized algorithm (ie, a suitable software). Such a device might have been invented independently of Jaskowski’s ideas, but a comparison with those ideas provides us with useful hints. So, before I report on a system belonging both to natural deduction and to Computer-Assisted Reasoning, CAR for short, it is worth while to recall Jaskowski’s precursory approach.

Both in the CAR system to be discussed and in Jaśkowski’s work there are no free variables (called also real variables) in predicate logic (by contrast with, eg, SB). In both systems this feature is related to the method of referencing rules. This method does not consist in putting rule names outside proof lines as metalinguistic comments; instead, one inserts suitable expressions inside the very text of the proof in question, as is usual in non-formalized, and in this sense natural, mathematical texts. Here is the relevant comment of Jaskowski as far as the concept of variable is concerned.

Symbols of variables which are not apparent variables do not merit the name of variable at all. We deal with such a term as with a given constant, though it is neither a primitive term nor a defined one. It is a constant the meaning of which, although undefined, remains unaltered through the whole process of reasoning. In practice, we often introduce such undefined constants in the course of a proof. For example we say: “Consider an arbitrary x” and then we deduce propositions which can be said to belong to the scope of constancy of the symbol x. This process of reasoning will be applied in our system. (p.28)

The procedure described above consists in the operation of eliminating a quantifier (what quantifier will be seen in the example discussed below), and this amounts to replacing a bound variable with a constant. However, instead of referencing a rule, one inserts an to the proof itself, viz the word “consider”. For proofs formalized in his system, Jaskowski suggests rendering this operation with the symbolic constant “T” which is analogous to the constant “S”, for “Suppose”, introduced previously (in propositional logic). We may guess that the letter “T” was chosen by the author to hint at the fact of substituting a term for a variable. The scope of constancy of the term “T” is called its domain; in this domain the meaning of the term is to be regarded as fixed. The author exemplifies his point with a proof of the following proposition (where “Û” stands for the universal quantifier, and “µ” for any propositional formula):

\[ CÛxÛyμxÛzμz. \]

Here is the proof.

(1) \( 1 \cdot S \ Cûxûy\mu x \)

Now we write "Tz" for "Consider an arbitrary z".

(2) \( 1 \cdot 1 \cdot Tz \)
(3) \( 1 \cdot 1 \cdot Ûyμz \)
(4) \( 1 \cdot 1 \cdot μz \)
(5) \( 1 \cdot Ûzμz \)
(6) \( Cûxûyμxûz \)

It is seen in this example that the constant "T" implicitly references the rule of eliminating the universal quantifier. The same can be done through explicit reference in the usual metalinguistic style, but Jaskowski’s method closer approximates the means of expression as usually adopted in non-formalized proofs; in this sense his formalization is more natural. Whether the word “consider” or another word is to be used for the reading of "T", this is a stylistic question which is irrelevant for the present discussion. Let it only be noted that the phrase “let x be so-and-so” may function as well to hint at the elimination of the universal quantifier, while “consider” in the context “consider x satisfying μ” seems to better express the existential quantifier elimination.

3. The issue of naturalness continued: a lesson of Euclid

It is worth examining the strategy of dealing with quantifiers as it appears in the paradigm of mathematical arguments, that is in The Elements of Euclid (quotations after I.Todhunter’s edition, re-edited by E.Rhys, London 1933). Let us take Proposition 1 in Book I. It states the following problem:

To describe an equilateral triangle on a given finite straight line.

This amounts to the statement prefixed with two quantifiers, viz:

For any finite straight line there exists an equilateral triangle that can be described on this line.
4. The name and what it stands for

For any line \( x \) there is a plane figure \( y \) such that:
\( y \) is an equilateral triangle, and \( y \) is described on \( x \).

In the proof of Proposition 1, which will be called "the thesis" below, the first sentence starts from the following assumption: Let \( AB \) be the given line. It is remarkable that the phrase "a given line", that means in the thesis any line whatever, switches to the phrase "the given line" in this starting assumption. This is a clear case of eliminating the existential quantifier in favour of an indefinite term (or, an undefined constant, in Jaskowski's terminology). In Euclid this operation occurs without any explicit reference to the rule of quantifier elimination, while the use of the phrase starting from "let" can be duly regarded as an implicit reference to this intuitive and general law of reasoning. In Jaskowski's system the same role is played by inserting the symbol "T" inside the relevant proof line. (Whether the Greek articles in Euclid's original do behave like the English articles in translations, is a problem for historical study; anyway, it seems reasonable to rely on the long tradition of interpreting Euclid, which guided the work of translators.)

Now we can see degrees of naturalness in formalizing a proof. Jaskowski's procedure is closer to that of Euclid and other mathematicians than the procedure, say, of the SB system; in this sense it becomes more natural. Then, if somebody provides us with such a formalization in which natural language expressions like "let" (instead of Jaskowski's "T") and "assume" (Jaskowski's "S") are employed, this formalization should be acknowledged as still more natural than that of Jaskowski. Such a highly natural formalization, at the same time being fully suited for a computer, is the main subject of the present discussion; it is to be presented in the next sections.

It is useful for the forthcoming discussion to look at some other facts in Euclid. In the proof in question, the quoted assumption "Let \( AB \) be the given straight line" is followed by the problem statement that can be read as an existencial statement since the problem consists in producing a required object, thus proving its existence; this runs as follows: "it is required to describe an equilateral triangle on \( AB \). The article prefixing "triangle" is again indefinite, as it was in the thesis, and thus the existential meaning remains preserved. Then the required triangle gets constructed and obtains the name \( ABC \). After it is constructed and so termed, its name deserves to be preceded with the definite article in the conclusion which reads as follows: "Wherefore the triangle \( ABC \) is equilateral, and it is described on the given straight line \( AB \)." Now both indefinite articles in the thesis in question, as stated at the start, are replaced with the definite articles, one before "line" and one before "triangle".

In such a statement of the conclusion, again some points regarding naturalness deserve a comment. A minor point consists in using Roman fonts for the word "wherefore" and italics for the conclusion itself; this typographical strategy, following that of Todhunter's edition, helps to distinguish between the content of the proof and such text junctures as the words "wherefore", "because", "but" appearing in the analyzed proof. A formalization of mathematical language that pretends to be as natural as possible should provide us with such vernacular junctures "wherefore".

The major point is concerned with the rule of generalization. Again, there is no explicit reference to this rule in the proof conclusion. But an implicit reference can be seen in the use of the definite article before "line". The history of dealing with the phrase "the line" in the course of the proof in question is sufficient reason to read "the line" as "every line"; the main part of this plot consists in using "let" in the moment of passing from the indefinite to the definite article.

Nothing of this kind happened to the predicate "triangle," and this is why the meaning of "the" as prefixing this predicate in the conclusion has to be different from the meaning of "the" in the phrase "the line". The first meaning, that attached above to "the line", can be duly called platonic for its frequent appearance in Plato, e.g. in The Republic, where "the" hints at an individual object taken as standing for a whole class; with the Schoolmen, this kind of use was called suppositio formalis. The second meaning of "the", in our example attached to "triangle", resembles the suppositio personalis of medieval logicians, and applies to individuals as individuals.

These considerations lead to the next postulate to be addressed to any formalization trying to be as natural as possible: it should use generalization in a way similar to that appearing in Euclid, that also in this respect has created a paradigm of text composition for mathematics. We shall see that also in this respect the system to be presented matches those high requirements, and succeeds in making computers able to assist natural inferences.

PART TWO: ON MIZAR MSE
Don’t try to guess what the name "Mizar" means. It was the author’s fancy to take a star’s name to stand for (i) a natural deduction system of (ii) Multi-Sorted predicate logic with Equality, for short MSE, (iii) that simulates the language of proofs, esp. that used by mathematicians, in a simplified and standardized form, adjusted to computer processing, and (iv) that is combined with a proof checker, ie. a program checking proof validity (checkers should be distinguished from provers; for the latter see Blaesius and Buercott 1987). Logical foundations for constructing provers and checkers have been prepared for by such theoretical results as those of Gentzen 1934-35, Herbrand 1930, Beth 1955; an instructive presentation of these achievements and their computer-oriented continuation is found in Robinson 1979.

Thus Mizar MSE, later on referred to as MSE, belongs to the class of systems devised for Computer-Assisted Reasoning, CAR for short. Both the logical system and the corresponding pieces of software are due to Andrzej Trybulec.

What distinguishes MSE from some other CAR systems is that it offers various modes of proof construction regarding text structure and vocabulary. This gives MSE linguistic flexibility, a feature welcomed by many users. What distinguishes MSE from traditional formalizations is that only premisses are explicitly referred to in justifying an inference, while inference rules, like in natural mathematical arguments, implicitly appear in a verbal context of symbolic expressions; e.g., "consider x such that" which, when preceding a formula, corresponds to the rule of existential quantifier elimination (the choice rule). Owing to these and other solutions in the Mizar language, the proofs written in it are more like ordinary mathematical texts than long and cumbersome formalizations in the Hilbert program style (see Hilbert and Bernays 1934). The price for these conveniences is that a richer vocabulary has to be mastered than that of ordinary predicate logic.

The MSE system as discussed in this comment does not pretend to suit the whole practice of reasoning in mathematics. Its use is restricted to elementary theories without functions. Note however, that multi-sortedness extends the possibilities of expression by introducing abstract objects as, so to say, abstract individuals, viz. sets, properties, relations, etc.

The presentation offered below is concerned with that version of MSE, used with a specially devised editor, which has its site at Warsaw University in Bialystok. It should be distinguished from that at the University of Alberta (also available through this issue). A use of the Bialystok software as made below is as detailed as it is necessary to grant an idea of how our proof checker works.

5. On what is obvious in proving

Any simulation of human reasoning involves the decision about what kinds of inferences are to be regarded as obvious. This is an important philosophical point. The contrary philosophical claim would be that of Descartes who seems to have believed that there are the smallest inferential steps, like atoms of logical evidence, which are shared by all human reasoners. A similar belief in atoms of evidence, though the atoms themselves are conceived of very differently, is found in Hilbert and Bernays' method of formalization. If a checker constructor does not acknowledge such a claim, then he takes advantage of freedom of choice among various admissible kinds of evidence.

The above philosophical point is crucial for understanding certain MSE features. This system arises from conscious decisions regarding the criteria of obviousness to be offered to system users. These criteria can be replaced with other ones, according to the user's supposed needs or the author's varying ideas. The user soon recognizes these features of MSE when at the very start he encounters a non-conventional approach. Contrary to textbook standards, MSE acknowledges all truth-functional tautologies as being obvious, independently of a degree of syntactic complexity. Furthermore, some predicate logic tautologies are regarded as obvious (see examples); in this point MSE differs from other formalized inferential systems, those in which every step should be justified by an explicit appeal to inference rules.

The fact that a formula is treated as obvious can in MSE be discovered in an experimental way. Namely, if the system gives its OK to a formula under proof without any intermediary steps, i.e. when one writes it down directly in the line following the title "proof" and precedes it with "thus", then the inference is to be recognized as obvious. This can be seen in Example 5:1. Note that "for x holds ..." corresponds to the universal quantifier, while "ex x st ..." ("st" for "satisfying") - to the existential quantifier.

```
environ
reserve x for integer;
A: not (for x holds P[x]);
begin
T:  ex x st (not P[x])
proof
thus thesis by A;
end;
```
Apart from parentheses, two punctuation devices are adopted: the colon separates the name of a formula from the formula in question, the semicolon ends every proof line (except for the thesis to be proven) and the whole proof. Double colons separate the proof itself, as controlled by the checker, from comments which may be added for the user’s convenience and are disregarded by the checker. The item "given" used in the premise forms a specially devised context to hint at an individual constant. MSE allows any letter string to be taken for a constant, and also any letter string for a variable, for these different functions are distinguishable owing to different contexts in MSE; however, for a better readability let a, b, c, a1, Socrates, etc. stand below for constants, and let x, y, z, x1, etc. stand for variables. This is illustrated in example 5:2.

```
environ
reserve x, Heisenberg for human;
given Heisenberg being human;
A: nobelprizewinner[Heisenberg];
begin
T: ex x st nobelprizewinner[x]
proof
thus thesis by A;
end;
```

Note, if we are not interested in specifying a particular sort, we can declare it as anything, or even use a dummy term, for instance "t"; the lack of such a type-specifying clause is treated in MSE as an error.

A kind of obviousness attaches to the axioms of identity, in the sense that they are "tacitly" built into the environ of each proof, and can be used without any referencing. There is a fairly wide literature on the notion of obviousness as relativized to various checkers; some data on this subject are found in Rudnicki 1987.

6. "Assume", "Let", "Given x such that"

As in the other inferential systems, when proving a conditional in MSE we assume the antecedent (hypothesis), or its part. This step is marked with the item assume; it does not require justification, i.e. referencing an (earlier proven) premise, for we may take advantage of a false hypothesis as well. (Note, what is to be proven it is not the consequent but the whole conditional, and this can be true while its antecedent happens to be false.)

The item let, specific for MSE, is the counterpart of the combined use both of elimination and of introduction of the universal quantifier, the latter usually being called generalization. The let statement is used to pick up and to fix a random object as representing all objects of a certain type. This object is denoted with a constant that will be generalized in a next step, the constant being included within the scope of for (the universal quantifier). Both assume and let constructions are shown in example 6:1.

```
environ
reserve a, y for t;
begin
T1: (for y holds P[y]) implies (ex y st P[y])
proof
assume 1: for y holds P[y];        :: 1 . SûyP(y)
now let a be t;                    :: 1 . 1 Ta
P[a] by 1;                         :: 1 . 1 P(a)
hence ex y st P[y];                :: 1 . ÉyP(y)
end;
hence thesis;                      :: CûyP(y)ÉyP(y)
end;
```

This example is to illustrate several points. There is a historical point hinted after the double colons (used in an editor for MSE to separate comments which are disregarded by the checker). The numbering follows Jaskowski’s method as exemplified in section 2 above to show the analogy to the MSE approach. "S" means "suppose" (in MSE "assume"); "1.1" is to introduce nesting (1.1 as nested in 1) which in MSE is rendered with "now" while "T" serves the purpose of introducing a fixed random term, and this is just that what "let" does in MSE.

As mentioned above, this text in a simple way exemplifies a nesting whose beginning is marked with "now" and the end with the word "end". The junction word “hence” has a role similar to that of “thus”, with the difference that it prefixes the statement which is justified by the immediately preceding line, and therefore the justifying reference (“by” etc.) is omitted. Such stylistic variations derive from MSE’s philosophy that natural modes of expression in reasonings should be imitated in computer formalization. When the antecedent of a conditional is an existential proposition, then the assumption of the antecedent
has to be accompanied by the existential quantifier elimination. This is done with the item "given a such that" (not to be mistaken for the construction "given a being" from example 5:2). Let this be shown in example 6:2.

### Example 6:2

```plaintext
environ
reserve a, x for t;
begin
T: (ex x st (p[x] & q[x])) implies ((ex x st p[x]) & (ex x st q[x]))
proof
given a such that 1: p[a] & q[a];
thus thesis by 1;
end;
```

This example gives us the opportunity to observe that our program permits the use of lower case letters instead of capitals in the role of predicates; at the same time it always requires brackets to follow predicates, even if no arguments are to be enclosed. The same example shows the obviousness of truth-functional tautologies and of existential generalization, as commented on above, in section 2. To obtain our thesis from 1, we need to split 1 into its two members (according to the truth-functional law of simplification), then to apply existential generalization (cf example 2) to each member, and then to join them according to the rule of introducing conjunction. None of these steps needs to be mentioned, as all of them are treated as obvious.

### 7. "Consider" (the Choice Rule)

If it is stated that an object satisfying certain conditions exists, then it is permissible to introduce such an object giving it a name. This principle is called the choice rule. When it is employed in a MSE text, its use is expressed with a "consider" statement including a constant, as it can be seen in example 7:1.

### Example 7:1

```plaintext
environ
reserve a, x for t;
begin
T: (ex x st p[x] & q[x]) implies ((ex x st p[x]) & (ex x st q[x]))
proof
assume A: ex x st p[x] & q[x];
B1: ex x st p[x]
proof
consider a such that A1: p[a] & q[a] by A;
B2: p[a] by A1;
thus thesis by B2;
end;
C1: ex x st q[x]
proof
consider a such that A1: p[a] & q[a] by A;
C2: q[a] by A1;
thus thesis by C2;
end;
thus thesis by B1, C1;
end;
```

Besides the use of "consider", this example shows that sometimes an auxiliary proof must be placed inside a larger argument; the former is what we call a nested proof. Nested proofs can be distinguished from the main proof by indenting, which makes them more readable; however, this device is not demanded by MSE. In the second nested proof (that of thesis C1) one could have used a different name, say "b", than the one used earlier ("a"), provided "b" has been reserved for this purpose in the type declaration in the environ.

A comparison of consider with let will be in order. The consider statement requires a justification while the let statement does not. It is so because the former is being asserted whereas the latter only supposed, and whether the supposition is true or false does not affect the truth or the falsity of the generalization to be proven. On the other hand, the application of the choice rule depends on the actual existence of the object being introduced, and therefore the relevant existential statement should be mentioned in the justification clause.

A useful application of consider is found in the indirect proofs of universal propositions and also of those conditionals which have a universal proposition in the consequent. Let the latter be exemplified with the following story - example 7:2.
environ
reserve x, Robin-Hood for bold;
A: for x holds (danger[] implies happy[x]);
begin
T: danger[] implies (for x holds happy[x])
proof
assume 1: not thesis;
2: danger[] by 1;
3: not (for x holds happy[x]) by 1;
consider Robin-Hood such that 4: not happy[Robin-Hood] by 3;
5: danger[] implies happy[Robin-Hood] by A;
6: happy[Robin-Hood] by 2,5;
thus contradiction by 4,6;
end;

Obviously, "danger" is short for "there-is-a-danger", and "happy" for "is-happy" (if written with such dashes, the phrases will be accepted by MSE's syntactic rules). Square brackets with empty content hint that the expression which they follow is a sentence (not a predicate, which needs arguments within the brackets). Such reasonings can be reproduced in MSE as well, so permissive is its syntax. It proves even more permissive due to some stylistic variants, some of them being meant as shortcuts.

Let the following be listed. Parentheses in A and T can be omitted. Premise A can be stated in a shorter form, resembling restricted quantification, viz "for x st danger[] holds happy[x]" (though this stylistic version does not seem lucky for this particular example). The conclusion can also run as follows: "hence contradiction by 4" which is to mean that line 6 is tacitly taken into account as the immediately preceding one, and therefore only the more remote line 4 is referenced. These are samples of MSE's linguistic flexibility which can be more and more extended, according to users' tastes.

8. Concluding remarks

Finally, let it be observed that MSE is not only a piece of software devised as a checker for an earlier existing system of predicate logic, as is the case, for instance, with TABLEAUX II. The latter, produced by Duncan Watt, Oxford University, functions like a software appendix to the system of analytic tableaux, esp. that found in Smullyan 1968 which lies in the Gentzen-Beth tradition. MSE is a system which contains a software integrated with a new logical language and a set of inference rules. The latter has some resemblances to Jaskowski’s approach but in some important points, such as multi-sortedness and the way of dealing with generalization, brings original solutions of its own. That such theoretical solutions in logic have been combined with a suitable software, gives this system useful applications in teaching predicate logic; see eg. A.Trybulec and Blair 1985 and 1985a, Mostowski and Z.Trybulec 1985, Zalewska 1987, Marciszewski 1987, Artalejo 1988, Malinowski 1990, Mostowski 1991 (the last published also in this issue, as a text for downloading).

A no less important field of educational application is found in teaching mathematics. This should be compared with the most advanced projects in this field, such as that of Patrick Suppes at Stanford, reported in Suppes 1981; see also Marinov 1983. Mizar applications in the computer-assisted teaching of mathematics are discussed, eg by Hoover and Rudnicki (this issue), Szczerba 1987. Mizar can also be applied to software verification; see, eg, Rudnicki and Drabent 1985.

A much wider range of applications attaches to the system called PC Mizar, devised by the same author Andrzej Trybulec supported by a number of collaborators from Polish and foreign universities. Its main ideas, including those related to Jaskowski, are exemplified in Mizar MSE, but this full PC Mizar system offers more powerful tools, as those for dealing with functions in mathematical proofs, etc, so that the whole of mathematics can be expressed in it. For authoritative and extensive information on the progress of the PC Mizar project - see the electronic Journal of Formalized Mathematics.

At the level of editing texts, a crucial difference between MSE and PC Mizar consists in the system of references functioning in the latter. While in MSE for each proof its premises have to be listed in the environ section, in PC Mizar one can take advantage of either axioms or proven theorems mentioned in other Mizar articles involved in the Mizar Library.

There are more systems aspiring to a similar universality as witnessed by a list of provers and proof checkers whose original site is at Stanford University. For a comparison, let me mention one (not included in the mentioned list) having been developed in Eindhoven, the Netherlands, under the name AUTOMATH. That system, unlike Mizar, is not based on classical predicate logic but on lambda calculus (see de Bruijn 1980, 1985). In the moment one cannot tell which approach is most likely to succeed in becoming a standard computer system for a worldwide net of collaboration in mathematics. Nevertheless, it seems certain that such a system should be created as a desirable device for
mathematical knowledge management. This idea wins ever wider appreciation. Its clear and convincing statement can be found in what is currently disseminated as QED Manifesto.

REFERENCES

General References


References concerning mechanized reasoning, esp MIZAR


reductio proof). A natural deduction system in pretty much the modern form was proposed by Gerhard Gentzen in his doctoral thesis of 1933. 1.2 What logic is about. In a significantly different way to Frege, Russell also thought of logic as science, as being in the business of systematizing the most general truths about the world.Â And reecting on that little suppositional inference, we see that a rule of inference governing âifâentitles us to assert if A and B, then A. If that is right, and if the point generalizes, then we just donât have to see such logical truths as reecting...
deep facts about the logical structure of the world (whatever that could mean): logical truth. Thus, we introduce the following Definition: The mean square error (MSE) of an estimator $\hat{\theta}$ of a parameter $\theta$, is the function of $\hat{\theta}$ dened by $E((\hat{\theta} - \theta)^2)$, and this is denoted as $MSE(\hat{\theta})$. This is also called the risk function of an estimator, with $(\hat{\theta} - \theta)^2$ called the quadratic loss function. The expectation is with respect to the random variables $X_1, \ldots, X_n$ since they are the only random components in the expression. Notice that the MSE measures the average squared difference between the estimator $\hat{\theta}$ and the parameter $\theta$, a somewhat reasonable measure of performance for an estimator. In general, any